Apollo: Lightweight Models for Dynamically Tuning Data-Dependent Code

DoE Centers of Excellence Performance Portability Meeting

David Beckingsale

April 20, 2016

LLNL-PRES-688701

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

What is Apollo?

- Emerging frameworks like RAJA & Kokkos allow portable code to be written with single source kernels
- This provides a mechanism for tuning the code, but no suggestions as to how to tune
- A code's behavior depends not only on the host architecture, but also on the input problem and run-time adaptation
- Apollo is an auto-tuning extension for RAJA that uses pre-trained, reusable modes to tune data-dependent code at runtime on a **kernel-by-kernel basis**

RAJA 101

```
forall<exec_policy>(iset, [&] (Index_type i)  {
     y[ i ] += a * x[ i ] ;
});
```

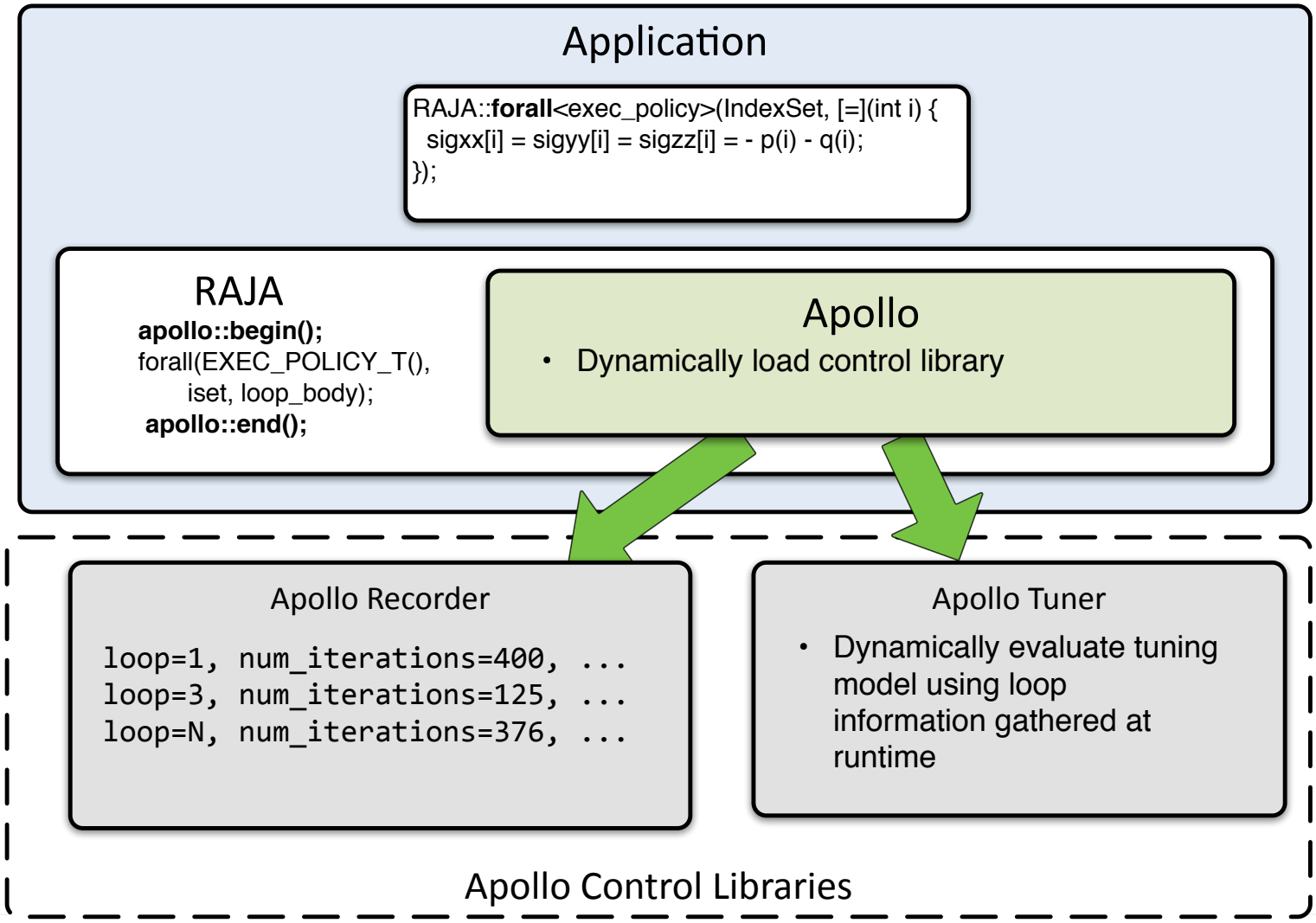
- Execution policy determines how loop iterations are scheduled to the hardware: OpenMP, Sequential, CUDA
- RAJA provides a native, C++ mechanism for tuning code at the kernel level, used in production DoE applications

- We focus on tuning the execution policy

Approach

- Traditional auto-tuners search a parameter space for the best configuration
- Using off-line training with statistical classifiers, we build lightweight decision models that **directly select values for tuning parameters**
- Our approach classifies kernels into categories where certain tuning parameters are “best”
- We use these classifiers to generate conditional statements (tuning models) that can be evaluated before each kernel execution
- These tuning models are low-overhead, and can respond quickly to changes in input data

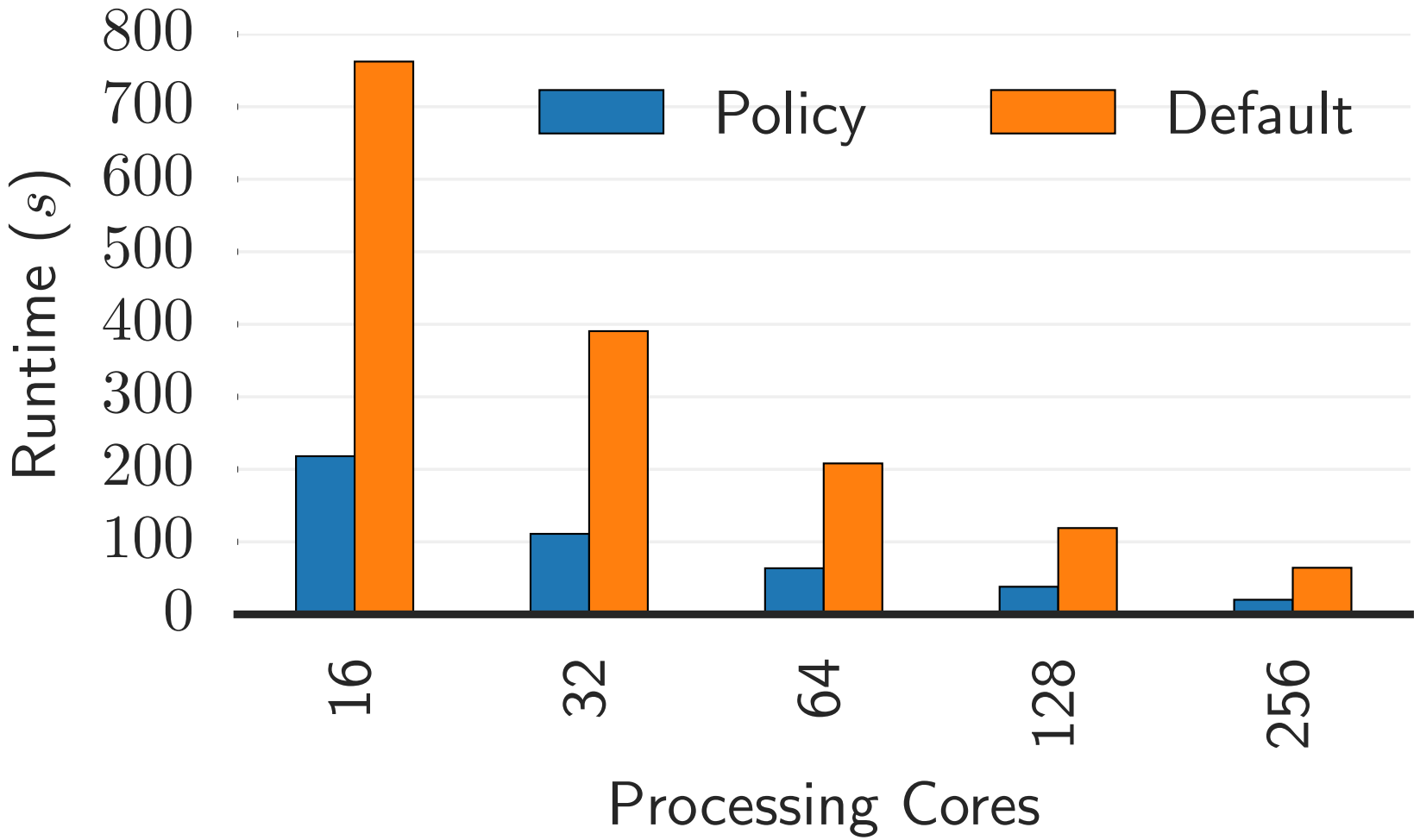
Apollo Workflow



Tuning CleverLeaf

- CleverLeaf is a hydrodynamics mini-application with Adaptive Mesh Refinement (AMR)
- AMR means that subdomains are created dynamically as the application runs, depending on the simulation
  - subdomain size is strongly correlated with the best execution policy
- Our models use kernel information, as well as application-level information like global problem size and current cycle count
- We apply the models on up to 16 nodes (256 cores) with each model tuning kernels executing on the subdomains local to a single MPI rank

Tuning CleverLeaf: up to 4.8x speedup



Limitations / What's next?

- Although our approach requires an offline training step, the models for CleverLeaf were generated with < 2 hours of training data and are re-usable across input decks
- We are working to make models that can tune multiple applications, allowing us to batch training data collection and amortize the upfront cost
- Apollo currently tunes execution policies on a homogeneous node, but with new CUDA 8.0 support for host/device lambdas, we can move to predicting **where** kernels should execute

Lawrence Livermore
National Laboratory

Implementation

- We add interface hooks to RAJA to allow Apollo to tune kernel parameters without recompilation
- Two control libraries can attach to these hooks:
  - Recorder: to record kernel information and runtime used to train our decision classifiers
  - Tuner: contains the conditional logic generated from the classifier to implement dynamic parameter selection
- The recorder collects information like iteration count, kernel size, and instruction counts
- The tuner evaluates the model logic and writes a predicted parameter value to a shared space where it used by the Apollo runtime to execute the kernel.